Mitigating Error Accumulation in Knowledge Editing for Multi-Hop Question Answering

Jiaxin Guo[1,2], **Hao Sun**[1,2], **Wenhao Zhang**[3], **Xuanbo Fan**[1,2], **Yan Zhang**[1,2*]

[1]School of Intelligence Science and Technology, Peking University
[2]State Key Laboratory of General Artificial Intelligence, Peking University, Beijing, China
[3]School of Computer Science, Shanghai Jiao Tong University
guojiaxin@stu.pku.edu.cn, zhyzhy001@pku.edu.cn

Abstract

Knowledge editing (KE) has emerged as an effective approach for updating factual information in large language models (LLMs) without the need for full retraining. Most of the existing methods for addressing the "ripple effect" in KE adopt a chain-structured reasoning process, making them vulnerable to error accumulation from early incorrect steps. Moreover, their conflict detection mechanisms are often susceptible to the LLM's inherent confirmation bias, further undermining the reliability of the editing process. To overcome these challenges, we propose **Tree of Editing (ToE), a tree-structured, retrieval-enhanced knowledge editing framework** designed to support robust reasoning under factual updates. ToE expands reasoning paths using a breadth-first strategy combined with score-guided beam search, enabling diverse and error-tolerant inference. Besides, we introduce an observer to objectively update knowledge, avoiding the bias caused by LLMs' over-confidence. Experimental results on two benchmarks, namely **MQuAKE-CF** (targeting ripple-aware editing) and **DUNE** (free-form editing), demonstrate that the ToE framework significantly outperforms existing methods.

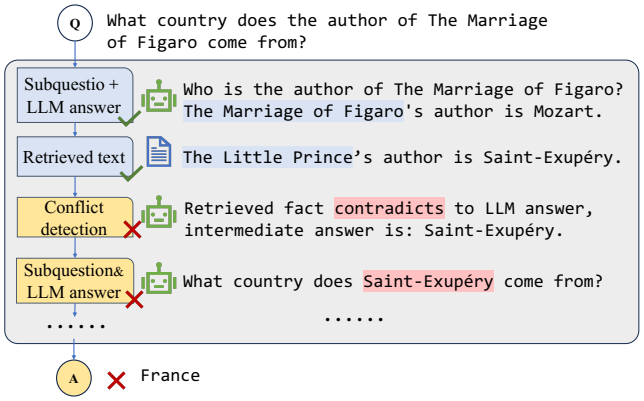

Figure 1: An illustration of cumulative error in chain-structured editing. The example shows how an initial error, stemming from contradictory judgments, leads to the generation of incorrect subsequent sub-questions and ultimately a wrong final answer.

1 Introduction

Large Language Models (LLMs) have achieved remarkable success across a wide range of natural language processing tasks (Zhao et al. 2023; Touvron et al. 2023; Achiam et al. 2023; Press et al. 2022). The factual knowledge embedded within these models tends to become outdated, posing a critical challenge for their sustained reliability and accuracy in real-world applications (Ilharco et al. 2022; Sinitsin et al. 2020). To address this limitation, the research community has proposed knowledge editing techniques (Sinitsin et al. 2020; De Cao, Aziz, and Titov 2021; Ni et al. 2023; Qi et al. 2024), which aim to update specific factual information in LLMs without training from scratch.

A key challenge in knowledge editing for LLMs is the **ripple effect**, where modifying a single factual statement requires corresponding updates to other related facts (Cohen et al. 2024; Zhong et al. 2023). It is usually evaluated by multi-hop question answering (MQA) (Khalifa et al. 2022; Rao, Mane, and Paliwal 2022). Although existing knowledge editing techniques have made significant progress in local updates, for example, some methods directly modify the exact neural activation during the LLM inference process (Meng et al. 2022a; Dai et al. 2021; Meng et al. 2022b), they often fall short in effectively handling the ripple effect via MQA.

An emerging direction to address the ripple effect is the editing of contextual knowledge using external memory modules (Zheng et al. 2023; Zhong et al. 2023). These methods have demonstrated higher editing success rates and improved scalability, along with enhanced robustness (Zhao et al. 2024; Wang et al. 2024a).

However, most contextual editing approaches follow a sequential decision-making paradigm (Zhong et al. 2023; Gu et al. 2023; Wang et al. 2024a). Once a knowledge update error occurs at a certain step, subsequent reasoning will deviate from the correct path, resulting in the **accumulation of errors** (LI et al. 2024; Bi et al. 2024), as illustrated in Figure 1. To address the challenge of error accumulation in multi-step reasoning, we naturally consider sampling multiple reasoning paths at each step to enhance the diversity of subproblems. This process gives rise to a tree structure, where we construct an evidence tree to enable multi-path exploration and conflict-aware knowledge editing—referred

*Corresponding author

to as **Tree of Edit (ToE)**. Specifically, at each reasoning step, the model generates multiple candidate sub-questions from the current node in a breadth-first manner and performs knowledge editing for each. Once a sufficient number of evidence nodes have been gathered, we aggregate the collected evidence to get the final answer.

Expanding a tree of depth k with n children per node incurs a computational complexity of $O(n^k)$, which is significantly higher than that of chain-based reasoning structures. To mitigate this issue, we propose a **Score-Guided Beam Expansion mechanism**, which dynamically filters high-quality nodes to limit the tree's width. This approach preserves the structural advantages of the tree while effectively reducing the time complexity to $O(n \cdot k \cdot beam\ size)$. In addition, the tree-based structure of ToE naturally supports parallel computation. By executing reasoning tasks along different paths concurrently on multiple GPUs, we can significantly reduce overall inference time, approaching the response speed of chain-based methods.

Moreover, most existing methods for identifying and resolving knowledge conflicts typically rely on the inherent capabilities of LLMs (Wang et al. 2024a). However, LLMs are prone to exhibit confirmation bias (Xu et al. 2024a), tending to be more faithful to information that aligns with their internal memory (Wang et al. 2024b). While fine-tuning offers a straightforward way to enhance conflict detection capabilities (Wang et al. 2024a), it requires task-specific annotated data and substantial computational resources (Wei et al. 2021; Mishra et al. 2021). To overcome these limitations, we introduce an auxiliary Observer module that objectively determines whether knowledge should be updated.

We evaluate ToE on multiple datasets, including the ripple-effect-sensitive dataset **MQuAKE-CF** and the free-form editing benchmark **DUNE**. Experimental results demonstrate that ToE achieves substantial improvements in editing reliability and reasoning generalization.

In short, our contributions include:

- We propose Tree of Editing (ToE), a novel tree-structured, retrieval-enhanced knowledge editing framework, which effectively mitigates the error accumulation problem inherent in chain-based reasoning methods.
- We develop a score-guided beam expansion strategy that balances the trade-off between reasoning diversity and computational efficiency, and introduce an observer to objectively judge conflicts, thus avoiding confirmation bias in LLM.
- Results on different datasets confirm that our ToE framework significantly enhances the factual adaptability of LLMs while maintaining robustness and efficiency.

2 Task Definition

2.1 Knowledge Editing

We consider a fact edit as the transformation of a factual triplet (s, r, o), into a modified one (s, r, o^*). This signifies an update to the object o associated with a given subject s and relation r, which we denote concisely as $e = (s, r, o \rightarrow o^*)$. Given a collection of such fact edits $\mathcal{E} = \{e_1, e_2, ...\}$ and an initial language model f, the primary goal of knowledge editing is to learn a function $F : f \times \mathcal{E} \rightarrow f^*$ that accurately reflect the changes introduced by the set of edits $\mathcal{E}$.

The quality of knowledge updating can be assessed through the following key indicators (Wang et al. 2024c). First, **reliability** refers to the updated model's accuracy on the new knowledge. For instance, the answer to "*Who is the President of the US*?" should change from "*Joe Biden*" to "*Donald Trump*". Second, **generalization** indicates the model's ability to correctly update semantically similar queries. Third, **locality** ensures that the update doesn't negatively impact the model's performance on unrelated questions.

2.2 Ripple Effect for Multi-hop Questions

Answering multi-hop edit questions necessitates the identification of a sequence of interconnected facts, forming a fact chain G_q. This chain starts with an initial head entity s_1 and progresses through a series of relationships $\{(s_1, r_1, o_1), (s_2, r_2, o_2), ..., (s_k, r_k, o_k)\}$, where the tail entity o_i of one fact serves as the head entity o_{i+1} for the subsequent fact. In the context of knowledge editing, the ripple effect (Cohen et al. 2024) refers to modifying a single fact within a chain can necessitate corresponding changes in subsequent linked facts to maintain logical coherence. For example, if the edit is $(UnitedStates, President, Biden \rightarrow Trump)$, the answer to "*Who is the spouse of the U.S. President?*" should also change accordingly. Accurately handling these ripple effects presents a greater challenge for LLMs.

3 Methodology

3.1 Overview

To answer queries with edit knowledge, we propose Tree of Editing (ToE), a novel framework that dynamically constructs an evidence tree through iterative branching and conflict-aware verification. As illustrated in Figure 2, given an input query Q, we initiate the query as the root node and hierarchically expand the tree by generating two distinct node types: Evidence nodes and Non-terminal Reasoning Nodes (Section 3.2). The latter includes a sub-question paired with a sub-answer, which derives from knowledge retrieval and an observer (Section 3.3). In addition, to enhance search efficiency and prevent the time overhead caused by expanding all nodes, we introduce a Score-Guided Beam Expansion mechanism (Section 3.4) that evaluates high-quality intermediate nodes for further expansion. When a sufficient number of evidence-answer nodes are generated, the final answer is obtained through weighted fusion of these nodes (the pseudocode is shown in Algorithm 1.

3.2 Evidence Tree Construction

ToE organizes the reasoning process as a breadth-first construction of an evidence tree. The root node N_0 corresponds to the original multi-hop query Q, and each expansion step aims to decompose complex reasoning into simpler subproblems. The construction process proceeds iteratively according to the following steps:

Evidence Check: For each node, we examine whether the current root-to-node path provides sufficient supporting

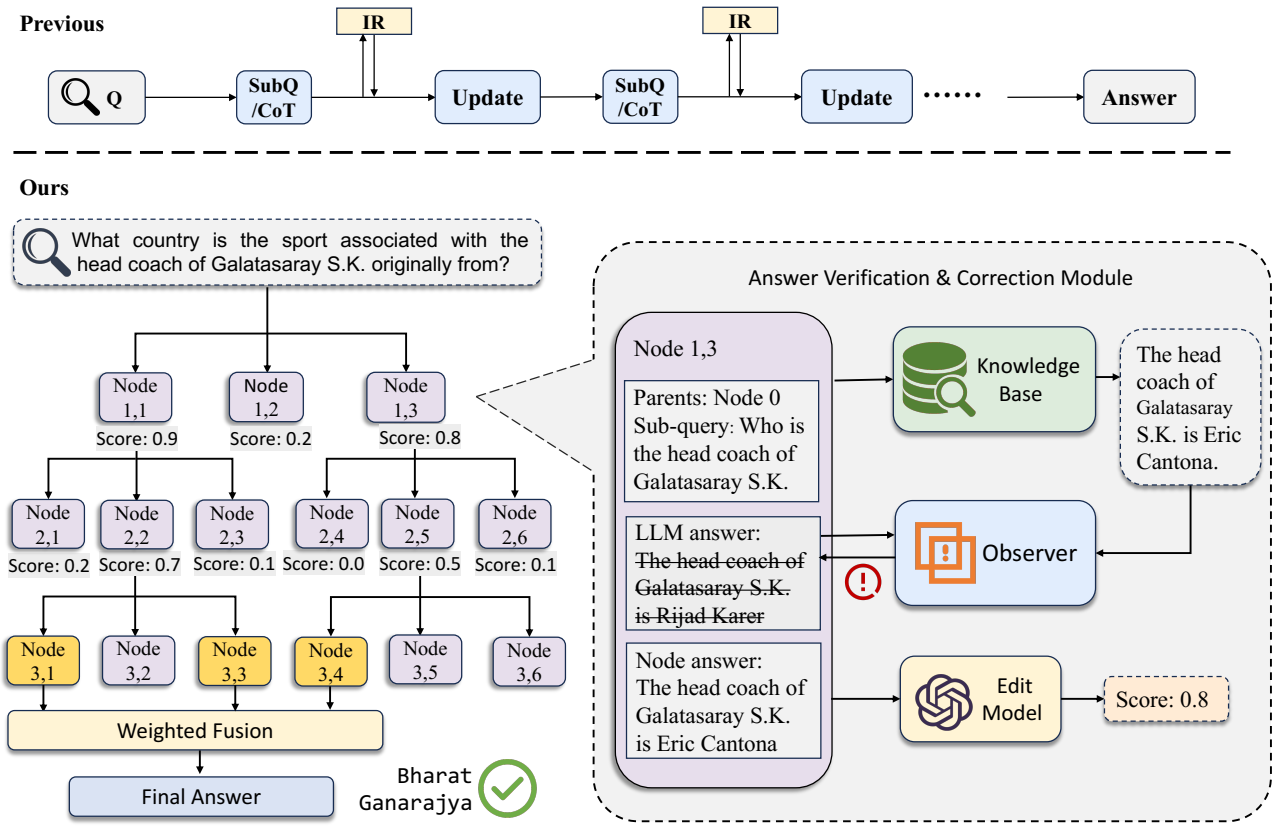

Figure 2: Overview of the Tree of Editing (ToE) framework. ToE expands a reasoning tree via sub-question generation. Each sub-question is answered by the model and verified using retrieved edit knowledge through an additional observer. Verified answers are scored for relevance and usefulness, and only top-scoring nodes are selected for further expansion. Once sufficient evidence nodes are gathered, the final answer is derived through weighted fusion of answers along the highest-confidence paths.

evidence to answer the original query Q. If the evidence is deemed sufficient, we terminate further expansion along this path and generate an Evidence Node. An evidence Node encapsulates the original query Q along with a local answer synthesized from the evidence accumulated along the path.

Reasoning Expansion: If the current evidence is insufficient, the model samples n candidate sub-questions from the current node. These sub-questions represent plausible intermediate reasoning steps. Each sampled sub-question forms a Non-terminal Reasoning Node, structured as a pair $\langle q_i, a_i^{\text{model}} \rangle$, where q_i is the generated sub-question and a_i^{model} is the model's initial attempt to answer it.

3.3 Conflict-Aware Edit Knowledge Injection

We inject edit knowledge in a conflict-aware manner. Following (Zhong et al. 2023), we adopt the Contriever retriever (Izacard et al. 2021) to store and retrieve edit knowledge. All edit knowledge is pre-encoded and stored in the external editing knowledge base. For each generated sub-question q_i, we retrieve the top-1 relevant sentence $a_i^{\text{retrieved}}$ from the editing knowledge base.

To determine whether the model's answer needs to be updated with the edited knowledge, we use an auxiliary observer to identify factual conflicts, avoiding the confirmation bias of LLMs. Specifically, compared to using LLMs as observers, we find that a lightweight Natural Language Inference (NLI) model offers a favorable trade-off between accuracy and inference efficiency (see ablation results in Section 6.3). The NLI-based observer is not only faster but also demonstrates competitive performance in detecting factual inconsistencies.

The Observer outputs a probability distribution over three labels: entailment, neutral, and contradiction. Specifically, we set $a_i^{\text{retrieved}}$ as the premise and a_i^{model} as the hypothesis. The final answer a_i^{final} for node i is determined as follows:

$$a_i^{\text{final}} = \begin{cases} a_i^{\text{model}}, & P(\text{neutral}) = \max(p) \\ a_i^{\text{retrieved}}, & \text{otherwise} \end{cases} \quad (1)$$

Where p denotes the output probability distribution over the three labels and $P(\text{neutral}) = \max(p)$ ensures that only when neutral is clearly the dominant label with high confidence does the model retain its original answer. In other cases, especially those where contradiction is likely or entailment is unclear, we defer to the retrieved knowledge.

This mechanism enables ToE to handle subtle conflicts and ambiguous entailments gracefully, improving the factual integrity of the reasoning process while preserving robustness

under noisy or uncertain conditions.

3.4 Score-Guided Beam Expansion

To mitigate the computational cost of exhaustively expanding all nodes in the evidence tree, we introduce a Score-Guided Beam Search mechanism that evaluates and selects the most promising nodes for further expansion.

After editing knowledge injection (Section 3.3), each reasoning node is assigned a finalized intermediate answer a_i^{final}. We then employ a scoring module to compute the score s_i for that node. The score is evaluated based on the following criteria: (1) semantic relevance between the sub-question of the node q_i and the original query Q, and (2) the constructiveness of the node's answer in contributing to the resolution of Q. Detailed scoring guidelines are provided in the appendix.

For all nodes at a certain depth, only the top-b scoring nodes (i.e., beam width b) are selected for expansion, effectively guiding the search process along high-quality reasoning trajectories. This reduces the computational burden compared to exhaustively expanding all nodes, lowering the complexity from exponential in the depth h (i.e., $O(n^h)$ for BFS) to a much more manageable $O(n \cdot h \cdot b)$.

Once the number of nodes of the collected evidence exceeds a threshold k, we perform a weighted response fusion to extract the final answer. Each evidence node contributes a candidate answer. For each answer, we compute the average score along the path from the root to the evidence node. The final answer is selected as the one associated with the highest average path score. Formally, let $\mathcal{P}_j$ denote the set of nodes along the path to the evidence node j, The final answer $Answer^{\text{final}}$ is given by:

$$Answer^{\text{final}} = \arg\max_j \left(\frac{1}{|P_j|} \sum_{i \in P_j} s_i \right) \tag{2}$$

Algorithm 1: Tree of Editing (ToE) Framework

Require: Query Q, Model M, KB K_{edit}, Beam b, Depth d, Evidence k, Branching n
Ensure: Final Answer A_{final}

```
Init root N_0 ← Q, E ← ∅, Queue ← [N_0]
while |E| < k and depth < d do
    NextQ ← ∅
    for N_i in Queue do
        if SUFFICIENT(N_i) then
            A_i ← EXTRACT(N_i), E ← E ∪ {(N_i, A_i)}
            continue
        end if
        Q ← GENSUBQ(N_i, n)
        for q_i ∈ Q do
            a^m ← M(q_i), a^r ← RETRIEVE(K_edit, q_i)
            p ← OBSERVE(a^m, a^r)
            a* ← argmax_p(a^m, a^r) based on formula 1
            s_i ← EVAL(q_i, a*, PATHTOROOT(i))
            NextQ ← NextQ ∪{(q_i, a*, s_i)}
        end for
    end for
    Queue ← TOPB(NextQ, b)
end while
Cands ← AGGREGATE(E)
A_final ← SELECTBEST(Cands)
return A_final =0
```

4 Experiments

4.1 Datasets and Metrics

We implement our method on two editing datasets: MQuAKE-CF (Zhong et al. 2023) and DUNE (Akyürek et al. 2023).**MQuAKE-CF** is constructed using fact triples from Wikidata (Vrandečić and Krötzsch 2014). This dataset evaluates LLMs' ability to handle counterfactual edits in multi-hop queries. Unlike prior datasets, MQuAKE-CF **assesses the ripple effect of multiple edits**, making it our primary benchmark.**DUNE** expresses edits in natural language rather than structured triples. It covers diverse domains: New Information and Scientific Reasoning.

MQuAKE-CF provides a **reliability metric**, measuring whether a model can answer post-edit multi-hop queries correctly. However, as discussed in Section 2.1, **generalization** is also a key dimension of knowledge editing. Thus we construct generalization queries for MQuAKE-CF, following the procedure detailed in appendix. For DUNE, which is a multiple-choice dataset, we use accuracy to evaluate **reliability** and **locality**. Examples of data formats and More details about the data can be found in the appendix.

4.2 Baselines

We compare our ToE framework with both parametric and non-parametric approaches. **Parametric methods** directly modify the model's weights to encode new information, such as Fine-Tuning (FT), Rank-One Model Editing (ROME) (Meng et al. 2022a), and Knowledge Neuron-based editing (KN) (Dai et al. 2021). In contrast, **non-parametric methods** preserve the original model weights and introduce edited knowledge at inference time, including MeLLo (Zhong et al. 2023), a memory-based method to retrieve factual edits; Retrieval-Augmented model editing (RAE) (Shi et al. 2024), which utilizes maximum mutual information to retrieve knowledge; PokeMQA (Gu et al. 2023), a programmable knowledge editing method for Multihop Question Answering; and EditCoT (Wang et al. 2024a), which constantly corrects CoT during the reasoning process. More implementation details about the baselines can be found in appendix .

4.3 Implementation Details

We evaluate all methods on two open-source LLMs: Meta-Llama-3-8B-Instruct (Grattafiori et al. 2024) and Qwen2.5-14B-Instruct (Yang et al. 2024). The NLI used in Section 3.3 is DeBERTa-v3-large-mnli-fever-anli-ling-wanli (He et al. 2020; Laurer 2022). Since the maximum number of reasoning hops in MQuAKE-CF is 4, we configure the evidence tree with a maximum depth of 5. Each reasoning node is allowed to expand up to $n = 3$ sub-nodes. We set the beam width b to 2 and terminate expansion once the number of collected evidence nodes reaches the threshold $k >= b$. More details about the baselines can be found in the appendix.

Model	Method	MQuAKE-CF			DUNE			
		Re	Ge	Avg	NI	NIL	Sci	Avg
Meta-Llama-3-8B -Instruct	Original Model	-	-	-	-	65.2	-	-
	ROME (Meng et al. 2022a)	4.0	0.0	2.0	69.1	58.4	81.4	69.6
	KN (Dai et al. 2021)	2.8	0.0	1.4	66.6	9.6	82.1	52.8
	FT (Hu et al. 2022)	3.6	1.2	2.4	30.3	23.5	11.1	21.6
	MeLLo (Zhong et al. 2023)	30.4	15.1	22.8	89.8	15.0	77.1	60.6
	PoKEMQA (Gu et al. 2023)	34.0	20.1	27.1	58.7	36.7	61.7	52.4
	EditCoT (Wang et al. 2024a)	35.4	16.0	25.7	91.3	59.7	**85.0**	78.7
	RAE-full (Shi et al. 2024)	43.7	21.6	32.7	-	-	-	-
	RAE-prune (Shi et al. 2024)	48.3	23.6	36.0	-	-	-	-
	ToE (Ours)	**53.1**	**27.9**	**40.5**	**93.3**	**61.4**	82.2	**79.0**
Qwen2.5-14B -Instruct	Original Model	-	-	-	-	64.4	-	
	ROME (Meng et al. 2022a)	0.0	0.0	0.0	73.1	60.1	75.5	69.6
	KN (Dai et al. 2021)	3.2	0.0	1.6	73.4	11.3	75.3	53.3
	FT (Hu et al. 2022)	9.5	1.7	5.6	76.2	25.2	13.8	38.4
	MeLLo (Zhong et al. 2023)	33.1	7.5	20.3	89.7	38.5	42.8	57.0
	PoKEMQA (Gu et al. 2023)	36.6	9.0	22.8	17.3	30.2	7.7	18.4
	EditCoT (Wang et al. 2024a)	34.2	11.3	22.8	93.8	61.0	86.3	80.4
	RAE-full (Shi et al. 2024)	32.9	7.8	20.4	-	-	-	-
	RAE-prune (Shi et al. 2024)	30.9	7.1	19.0	-	-	-	-
	ToE (Ours)	**38.9**	**13.9**	**26.4**	**94.0**	**63.1**	**87.2**	**81.4**

Table 1: Accuracy Comparison of Different Methods on MQuAKE-CF and DUNE Datasets. Re: Reliability, Ge: Generalization, Avg: Average, NI: New Info, NIL: New Info Locality, Sci: Science. For the New Info Locality metric, to compare with the unedited model, we tested the accuracy of the original model. RAE cannot be applied to the DUNE because DUNE cannot construct the knowledge graphs required for RAE.

5 Overall Performance

5.1 Effectiveness of Knowledge Editing

We compare ToE with a range of knowledge editing baselines across the MQuAKE-CF and DUNE datasets. The results are presented in Table 1. Compared to the baseline methods, our approach achieves promising results.

First, parameter-based knowledge editing methods exhibit very low editing accuracy, with only 4.0 for ROME and 2.8 for KN. This is because such approaches often fail to scale to large-scale editing scenarios - extensive modifications to model parameters tend to destabilize the model's output behavior, which is consistent with the findings reported in (Li et al. 2024). In contrast, our method does not modify model parameters, preserving the original knowledge and capabilities of the model and enabling more accurate reflection of injected counterfactual facts during the reasoning process.

Second, our method also outperforms existing in-context editing techniques. We attribute this improvement to the proposed conflict-aware tree-based search mechanism, which enables the model to explore more diverse and higher-quality reasoning paths. Unlike linear or retrieval-based editing strategies, our tree-structured exploration helps model avoid error propagation and maintain logical consistency throughout the reasoning process.

Moreover, we evaluated the accuracy across different reasoning hops, as shown in Figure 3. It can be observed that our method demonstrates stronger reasoning capability and greater robustness against performance degradation on high-hop questions. Notably, some baseline methods (e.g., MeLLo) exhibit a sharp decline in accuracy as the number of reasoning hops increases, further highlighting the stability of our method in deeper multi-hop reasoning scenarios.

	PokeMQA	MeLLo	RAE	EditCoT	ToE
GPT-4o-mini	35.3	43.2	55.3	33.5	54.1
GPT-4o	38.0	45.3	58.7	45.0	60.8

Table 2: Editing reliability on proprietary black-box LLMs evaluated on the MQuAKE-CF dataset. Our method performs well across diverse LLMs.

5.2 Generalization and Locality Evaluation

As shown in Table 1, the generalization metric shows a significant performance drop compared to the reliability metric. We attribute this to longer contexts in generalization queries, and reasoning over longer contexts is more challenging, which aligns with the findings of (Xu et al. 2024b). Nevertheless, our method still demonstrates superior performance, demonstrating that our ToE architecture offers stronger capabilities for complex reasoning tasks.

Although all methods incur a drop in locality compared to the original model (65.2 and 64.4), our method achieves the closest performance (61.4 and 63.1). This indicates ToE introduces minimal side effects on unedited queries.

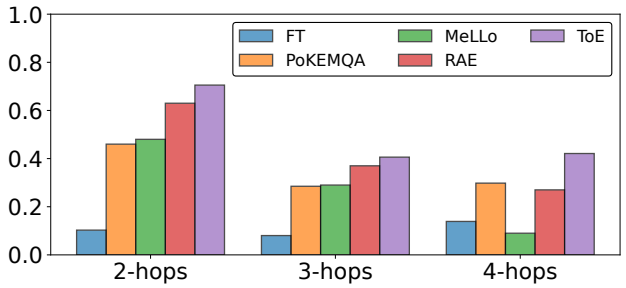

Figure 3: Editing accuracy across different hop counts on reliability of MQuAKE-CF. ToE maintains high accuracy even as hop count increases.

6 Analysis

6.1 Editing Accuracy on Proprietary Black-box Models

To investigate the compatibility of our method across language models of different types and scales, we evaluate the editing performance on the closed-source GPT model using the MQuAKE-CF dataset, as shown in Table 2. Since the retrieval phase of RAE requires access to the model's output probability distribution, we adopt the open-source Llama3-8B as a proxy model. The results show that our method not only performs well on open-source LLMs, but also generalizes effectively to large closed-source models, demonstrating the robustness and flexibility of our method.

6.2 Hyperparameter Sensitivity

We conduct sensitivity analysis on two critical factors: beam width b and branching factor n. Table 3 reports the editing accuracy under different combinations of these parameters.

We observe that increasing the beam width b and the number of node expansions n initially leads to substantial improvements in overall performance, as reflected by the AVG metric, indicating that expanding the search space indeed allows the model to capture more informative reasoning paths. However, further increasing the hyperparameters to $b = 3, n = 5$ only yields a slight improvement, and even slightly reduces performance on reliability of MQuAKE-CF. This suggests that while broader search somewhat increases the chance of discovering informative paths, it also introduces more noise and computational cost.

These results highlight a non-linear relationship between search scope and model accuracy. This implies that using moderate values such as $b = 2, n = 3$ strikes a good balance between reasoning accuracy and computational efficiency.

6.3 Ablation Study

We perform ablation studies on two modules: (1) replacing the NLI-based observer with different type of LLMs for conflict detection, and (2) expanding the lowest-scoring instead of the top-scoring nodes during reasoning. As shown in Table 4, replacing the observer with small LLMs leads to clear performance drops (down to 38.1), while larger LLMs incur high latency. Our method achieves a better balance between accuracy and efficiency. Similarly, expanding low-scoring nodes severely degrades performance (down to 23.4), confirming that prioritizing high-quality reasoning paths is critical. These results underscore the necessity of both components for effective and robust multi-hop knowledge editing.

	Re	NI	SCI	AVG
$b = 1, n = 1, m = 5$	42.0	74.2	68.9	61.7
$b = 1, n = 2, m = 5$	45.9	83.0	78.1	69.0
$b = 2, n = 3, m = 5$	53.1	93.3	82.2	76.2
$b = 3, n = 5, m = 5$	52.7	94.0	84.4	77.0

Table 3: Hyperparameter sensitivity on different dataset metrics. The rate of accuracy improvement slows down with higher hyperparameter values.

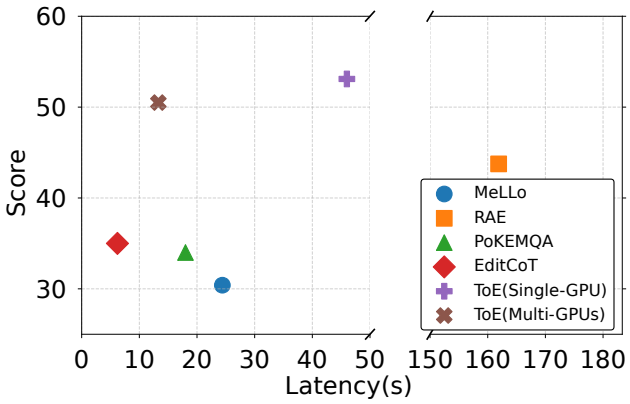

Figure 4: Relationship between inference time and performance. Our method achieves better time efficiency by leveraging multi-GPU parallelism.

6.4 Efficiency Analysis

Tree-structured reasoning often increases computational overhead due to node expansion. To address this, ToE supports dynamic adjustment of expansion breadth and leverages parallelism to offset the added cost. We evaluate two settings on MQuAKE-CF with Meta-Llama-3-8B-Instruct over 300 examples. In the Single-GPU setup, ToE runs on one A800-80G GPU, matching baselines. In the Multi-GPU setup, four A800 GPUs each explore a tree branch in parallel, with asynchronous result collection.

Figure 4 shows the trade-off between latency and accuracy. While ToE incurs higher latency than chain-based methods (e.g., MeLLo, PokeMQA) under a single GPU due to its tree expansion, the increase is moderate—only 2–3× despite a width-6 structure—making it a practical design. Crucially, ToE's core operations (node expansion and scoring) are highly parallelizable. With Multi-GPU execution, ToE reduces latency to near chain-based levels, demonstrating strong scalability and efficient hardware utilization. In contrast, RAE suffers from much higher latency due to its reliance on querying both a knowledge graph and external sources such as Wikidata via API calls.

Method	Type	LLM Type	Expansion	MQuAKE	DUNE	OA	OL(s)
ToE (Full)	observer-based	-	Top-b Score	53.1	93.3	92.0	0.09
Ablation 1	LLM-based	Llama-3-8B-Instruct	Top-b Score	38.1	78.2	78.5	4.25
Ablation 2	LLM-based	GPT-4o-mini	Top-b Score	51.8	93.7	91.5	1.58
Ablation 3	LLM-based	Qwen2.5-32B-Instruct	Top-b Score	49.9	89.8	87.0	8.81
Ablation 4	observer-based	-	Bottom-b Score	23.4	81.0	91.8	0.09

Table 4: Comparison of different observer types and expansion strategies. OA: Observer Accuracy, OL: Observer Latency.

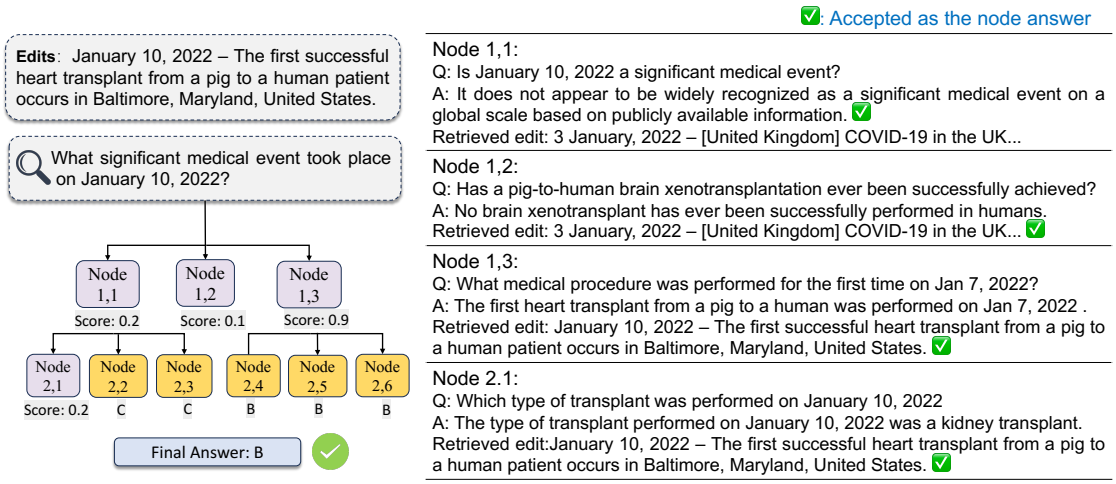

Figure 5: A case study of our ToE framework on the DUNE Dataset

6.5 Case Study

Figure 5 illustrates a case of ToE's reasoning on the one-hop DUNE dataset, and complete case studies on other datasets are provided in the appendix. We can observe that the ToE method selects and expands only the nodes that contribute the most to solving the problem, while avoiding the expansion of low-quality nodes (such as those that are repetitive, irrelevant, or ambiguous). When faced with complex multi-hop questions, our method also demonstrates significant advantages.

7 Related Work

Previous research has investigated various methods for editing LLM knowledge (Wang et al. 2024a, 2025; Zhang et al. 2024). Some of these methods focus on locating and modifying the model parameters associated with specific concepts. (Meng et al. 2022a) used causal intervention analysis and Rank One model editing methods to locate and modify specific intermediate layer MLP modules in the GPT model. (Meng et al. 2022b) implemented large-scale memory editing in the Transformer model. To address the challenges of locality and generality more systematically, global optimization strategies have been proposed (Yu et al. 2024; Lee et al. 2022; Mitchell et al. 2021; Hase et al. 2023; Ni et al. 2023). Recent research has demonstrated the enhanced knowledge editing capabilities of in-context editing (Zheng et al. 2023). (Zhong et al. 2023) stores the edited facts in explicit memory and gradually decomposes the multi-hop problem during the reasoning process. (Shi et al. 2024) achieved multi hop question answering knowledge editing through fact retrieval based on mutual information maximization. Unlike previous approaches (Pinter and Elhadad 2023; Yao et al. 2023; Zheng et al. 2025), we introduce a novel method that effectively mitigates the error accumulation issue in existing techniques. Our approach organizes the reasoning process hierarchically, enabling more accurate and robust knowledge editing.

8 Conclusion

In this paper, we introduce Tree of Editing (ToE), a novel framework for multi-hop knowledge editing that leverages a tree-structured reasoning paradigm. Unlike prior chain-based approaches that are prone to error accumulation, ToE performs breadth-first expansion to construct an evidence tree, where each node is evaluated through a conflict-aware verification mechanism. By integrating retrieved edit knowledge using an observer and guiding the expansion process with node scoring and beam selection, ToE enables diverse and parallel reasoning paths. Experimental results on MQuAKE-CF and DUNE demonstrate that ToE consistently outperforms both parametric and non-parametric baselines across multiple editing metrics, including reliability, generalization, and locality. Further analysis on closed-source models and ablation studies confirms the effectiveness and generality of our framework. These results highlight the potential of tree-based reasoning in improving factual consistency and adaptability in edited language models.

Acknowledgments

This work is supported in part by Ucap Cloud.

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Akyürek, A. F.; Pan, E.; Kuwanto, G.; and Wijaya, D. 2023. Dune: Dataset for unified editing. *arXiv preprint arXiv:2311.16087*.

Bi, Z.; Hajialigol, D.; Sun, Z.; Hao, J.; and Wang, X. 2024. STOC-TOT: Stochastic Tree-of-Thought with Constrained Decoding for Complex Reasoning in Multi-Hop Question Answering. *arXiv preprint arXiv:2407.03687*.

Cohen, R.; Biran, E.; Yoran, O.; Globerson, A.; and Geva, M. 2024. Evaluating the ripple effects of knowledge editing in language models. *Transactions of the Association for Computational Linguistics*, 12: 283–298.

Dai, D.; Dong, L.; Hao, Y.; Sui, Z.; Chang, B.; and Wei, F. 2021. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*.

De Cao, N.; Aziz, W.; and Titov, I. 2021. Editing factual knowledge in language models. *arXiv preprint arXiv:2104.08164*.

Grattafiori, A.; Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Vaughan, A.; et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Gu, H.; Zhou, K.; Han, X.; Liu, N.; Wang, R.; and Wang, X. 2023. Pokemqa: Programmable knowledge editing for multi-hop question answering. *arXiv preprint arXiv:2312.15194*.

Hase, P.; Diab, M.; Celikyilmaz, A.; Li, X.; Kozareva, Z.; Stoyanov, V.; Bansal, M.; and Iyer, S. 2023. Methods for measuring, updating, and visualizing factual beliefs in language models. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, 2714–2731.

He, P.; Liu, X.; Gao, J.; and Chen, W. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W.; et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2): 3.

Ilharco, G.; Ribeiro, M. T.; Wortsman, M.; Gururangan, S.; Schmidt, L.; Hajishirzi, H.; and Farhadi, A. 2022. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*.

Izacard, G.; Caron, M.; Hosseini, L.; Riedel, S.; Bojanowski, P.; Joulin, A.; and Grave, E. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

Khalifa, M.; Logeswaran, L.; Lee, M.; Lee, H.; and Wang, L. 2022. Few-shot reranking for multi-hop qa via language model prompting. *arXiv preprint arXiv:2205.12650*.

Laurer, M. 2022. deberta-v3-large-mnli-fever-anli-ling-wanli. https://huggingface.co/MoritzLaurer/deberta-v3-large-mnli-fever-anli-ling-wanli.

Lee, K.; Han, W.; Hwang, S.-w.; Lee, H.; Park, J.; and Lee, S.-W. 2022. Plug-and-play adaptation for continuously-updated qa. *arXiv preprint arXiv:2204.12785*.

LI, J.; Liu, R.; Li, Y.; Zhou, T.; Li, M.; and Chen, X. 2024. Tree of Reviews: A Tree-based Dynamic Iterative Retrieval Framework for Multi-hop Question Answering. *arXiv preprint arXiv:2404.14464*.

Li, Q.; Liu, X.; Tang, Z.; Dong, P.; Li, Z.; Pan, X.; and Chu, X. 2024. Should We Really Edit Language Models? On the Evaluation of Edited Language Models. *arXiv preprint arXiv:2410.18785*.

Meng, K.; Bau, D.; Andonian, A.; and Belinkov, Y. 2022a. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35: 17359–17372.

Meng, K.; Sharma, A. S.; Andonian, A.; Belinkov, Y.; and Bau, D. 2022b. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*.

Mishra, S.; Khashabi, D.; Baral, C.; and Hajishirzi, H. 2021. Cross-task generalization via natural language crowdsourcing instructions. *arXiv preprint arXiv:2104.08773*.

Mitchell, E.; Lin, C.; Bosselut, A.; Finn, C.; and Manning, C. D. 2021. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*.

Ni, S.; Chen, D.; Li, C.; Hu, X.; Xu, R.; and Yang, M. 2023. Forgetting before learning: Utilizing parametric arithmetic for knowledge updating in large language models. *arXiv preprint arXiv:2311.08011*.

Pinter, Y.; and Elhadad, M. 2023. Emptying the Ocean with a Spoon: Should We Edit Models? *arXiv preprint arXiv:2310.11958*.

Press, O.; Zhang, M.; Min, S.; Schmidt, L.; Smith, N. A.; and Lewis, M. 2022. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*.

Qi, S.; Yang, B.; Jiang, K.; Wang, X.; Li, J.; Zhong, Y.; Yang, Y.; and Zheng, Z. 2024. In-context editing: Learning knowledge from self-induced distributions. *arXiv preprint arXiv:2406.11194*.

Rao, D. J.; Mane, S. S.; and Paliwal, M. A. 2022. Biomedical multi-hop question answering using knowledge graph embeddings and language models. *arXiv preprint arXiv:2211.05351*.

Shi, Y.; Tan, Q.; Wu, X.; Zhong, S.; Zhou, K.; and Liu, N. 2024. Retrieval-enhanced knowledge editing in language models for multi-hop question answering. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 2056–2066.

Sinitsin, A.; Plokhotnyuk, V.; Pyrkin, D.; Popov, S.; and Babenko, A. 2020. Editable neural networks. *arXiv preprint arXiv:2004.00345*.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.;

Azhar, F.; et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Vrandečić, D.; and Krötzsch, M. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10): 78–85.

Wang, C.; Su, W.; Ai, Q.; and Liu, Y. 2024a. Knowledge Editing through Chain-of-Thought. *arXiv preprint arXiv:2412.17727*.

Wang, F.; Wan, X.; Sun, R.; Chen, J.; and Arık, S. Ö. 2024b. Astute rag: Overcoming imperfect retrieval augmentation and knowledge conflicts for large language models. *arXiv preprint arXiv:2410.07176*.

Wang, P.; Tang, Z.; Zhou, K.; Li, J.; Zhu, Q.; and Zhang, M. 2025. Revealing and mitigating over-attention in knowledge editing. *arXiv preprint arXiv:2502.14838*.

Wang, S.; Zhu, Y.; Liu, H.; Zheng, Z.; Chen, C.; and Li, J. 2024c. Knowledge editing for large language models: A survey. *ACM Computing Surveys*, 57(3): 1–37.

Wei, J.; Bosma, M.; Zhao, V. Y.; Guu, K.; Yu, A. W.; Lester, B.; Du, N.; Dai, A. M.; and Le, Q. V. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Xu, R.; Qi, Z.; Guo, Z.; Wang, C.; Wang, H.; Zhang, Y.; and Xu, W. 2024a. Knowledge conflicts for llms: A survey. *arXiv preprint arXiv:2403.08319*.

Xu, X.; Xiao, T.; Chao, Z.; Huang, Z.; Yang, C.; and Wang, Y. 2024b. Can LLMs Solve longer Math Word Problems Better? *arXiv preprint arXiv:2405.14804*.

Yang, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Li, C.; Liu, D.; Huang, F.; Wei, H.; et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Yao, Y.; Wang, P.; Tian, B.; Cheng, S.; Li, Z.; Deng, S.; Chen, H.; and Zhang, N. 2023. Editing large language models: Problems, methods, and opportunities. *arXiv preprint arXiv:2305.13172*.

Yu, L.; Chen, Q.; Zhou, J.; and He, L. 2024. Melo: Enhancing model editing with neuron-indexed dynamic lora. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 19449–19457.

Zhang, N.; Yao, Y.; Tian, B.; Wang, P.; Deng, S.; Wang, M.; Xi, Z.; Mao, S.; Zhang, J.; Ni, Y.; et al. 2024. A comprehensive study of knowledge editing for large language models. *arXiv preprint arXiv:2401.01286*.

Zhao, W. X.; Zhou, K.; Li, J.; Tang, T.; Wang, X.; Hou, Y.; Min, Y.; Zhang, B.; Zhang, J.; Dong, Z.; et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2).

Zhao, Z.; Yang, Y.; Li, Y.; and Cao, Y. 2024. RIPPLECOT: Amplifying Ripple Effect of Knowledge Editing in Language Models via Chain-of-Thought In-Context Learning. *arXiv preprint arXiv:2410.03122*.

Zheng, C.; Li, L.; Dong, Q.; Fan, Y.; Wu, Z.; Xu, J.; and Chang, B. 2023. Can we edit factual knowledge by in-context learning? *arXiv preprint arXiv:2305.12740*.

Zheng, J.; Qiu, S.; Shi, C.; and Ma, Q. 2025. Towards lifelong learning of large language models: A survey. *ACM Computing Surveys*, 57(8): 1–35.

Zhong, Z.; Wu, Z.; Manning, C. D.; Potts, C.; and Chen, D. 2023. Mquake: Assessing knowledge editing in language models via multi-hop questions. *arXiv preprint arXiv:2305.14795*.